Try 36 hard

May 6, 2025

[1]:
```
import os
import random
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import backend as K
from tensorflow.keras import layers, Model
from tensorflow.keras.utils import plot_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_curve, auc, accuracy_score
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,␣
↪ModelCheckpoint
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

# Seeds
random.seed(42)
np.random.seed(42)
tf.random.set_seed(42)

# Configuration
IMG_SIZE = (128, 128)
BATCH_SIZE = 32
EPOCHS = 150
DATA_PATH = "/kaggle/input/socofing/SOCOFing/Real/"
ALTER_HARD_PATH = "/kaggle/input/socofing/SOCOFing/Altered/Altered-Hard/"

# Data Loading and Preprocessing
def load_fingerprint_data(data_path):
    images = []
    labels = []

    for filename in os.listdir(data_path):
        if filename.endswith(".BMP"):
            parts = filename.split('__')
            person_id = parts[0]
            finger_info = parts[1].split('_')
```

```
            hand = finger_info[1]
            finger_type = finger_info[2]
            label = f"{person_id}_{hand}_{finger_type}"

            img_path = os.path.join(data_path, filename)
            img = load_img(img_path, color_mode='grayscale',␣
 ↪target_size=IMG_SIZE)
            img = img_to_array(img).astype('float32') / 255.0
            images.append(img)
            labels.append(label)

    images = np.array(images)
    labels = np.array(labels)

    if images.ndim != 4:
        raise ValueError(f"Expected 4D array (num_samples, height, width,␣
 ↪channels), got {images.shape}")

    return images, labels

# Load both datasets
images_real, labels_real = load_fingerprint_data(DATA_PATH)
images_alter_hard, labels_alter_hard = load_fingerprint_data(ALTER_HARD_PATH)
print(f"Loaded {len(images_real)} images from Real dataset")
print(f"Loaded {len(images_alter_hard)} images from Alter-hard dataset")
print(f"Image shape: {images_real[0].shape}")

# Pair Generation
def create_pairs(images_real, labels_real, images_alter_hard,␣
 ↪labels_alter_hard):
    label_to_real_image = {}
    for img, label in zip(images_real, labels_real):
        label_to_real_image[label] = img

    label_to_alter_images = {}
    for img, label in zip(images_alter_hard, labels_alter_hard):
        if label not in label_to_alter_images:
            label_to_alter_images[label] = []
        label_to_alter_images[label].append(img)

    # Generate positive pairs
    positive_pairs = []
    for label in label_to_real_image:
        if label in label_to_alter_images:
            real_img = label_to_real_image[label]
            for alter_img in label_to_alter_images[label]:
                positive_pairs.append([real_img, alter_img])
```

```
    # Generate negative pairs
    negative_pairs = []
    num_positive = len(positive_pairs)

    common_labels = list(set(label_to_real_image.keys()) &␣
 ↪set(label_to_alter_images.keys()))

    while len(negative_pairs) < num_positive:
        label1 = random.choice(common_labels)
        label2 = random.choice(common_labels)
        if label1 != label2:
            real_img = label_to_real_image[label1]
            alter_img = random.choice(label_to_alter_images[label2])
            negative_pairs.append([real_img, alter_img])

    # Combine and shuffle pairs
    pairs = positive_pairs + negative_pairs
    pair_labels = [1] * len(positive_pairs) + [0] * len(negative_pairs)

    indices = np.arange(len(pairs))
    np.random.shuffle(indices)
    pairs = np.array(pairs)[indices]
    pair_labels = np.array(pair_labels)[indices]

    return pairs, pair_labels

pairs, labels = create_pairs(images_real, labels_real, images_alter_hard,␣
 ↪labels_alter_hard)
print(f"Generated {len(pairs)} pairs")

# Train/Validation/Test Split
X_train, X_temp, y_train, y_temp = train_test_split(
    pairs, labels, train_size=0.7, test_size=0.3, random_state=42,␣
 ↪stratify=labels
)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, train_size=0.5, test_size=0.5, random_state=42,␣
 ↪stratify=y_temp
)

print(f"Training pairs: {len(X_train)}")
print(f"Validation pairs: {len(X_val)}")
print(f"Test pairs: {len(X_test)}")
print(f"Training class distribution: {np.bincount(y_train)}")
print(f"Validation class distribution: {np.bincount(y_val)}")
```

```
print(f"Test class distribution: {np.bincount(y_test)}")
```

```
2025-05-05 22:51:31.155459: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:477] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
E0000 00:00:1746485491.178513   37181 cuda_dnn.cc:8310] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1746485491.185549   37181 cuda_blas.cc:1418] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
```

```
Loaded 6000 images from Real dataset
Loaded 14272 images from Alter-Easy dataset
Image shape: (128, 128, 1)
Generated 28544 pairs
Training pairs: 19980
Validation pairs: 4282
Test pairs: 4282
Training class distribution: [9990 9990]
Validation class distribution: [2141 2141]
Test class distribution: [2141 2141]
```

[2]:
```
# Siamese Network Architecture
def create_embedding_network(input_shape):
    inputs = layers.Input(shape=input_shape)
    x = layers.Conv2D(32, (5, 5), activation='relu', kernel_regularizer=tf.
 ↪keras.regularizers.l2(0.01))(inputs)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv2D(64, (3, 3), activation='relu', kernel_regularizer=tf.
 ↪keras.regularizers.l2(0.01))(x)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv2D(128, (3, 3), activation='relu', kernel_regularizer=tf.
 ↪keras.regularizers.l2(0.01))(x)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.BatchNormalization()(x)

    x = layers.Conv2D(256, (3, 3), activation='relu', kernel_regularizer=tf.
 ↪keras.regularizers.l2(0.01))(x)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.BatchNormalization()(x)
```

```
    x = layers.Conv2D(512, (3, 3), activation='relu', kernel_regularizer=tf.
 ↪keras.regularizers.l2(0.01))(x)
    x = layers.MaxPooling2D(pool_size=(2, 2))(x)
    x = layers.BatchNormalization()(x)

    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(256, activation='relu', kernel_regularizer=tf.keras.
 ↪regularizers.l2(0.01))(x)
    x = layers.Dropout(0.35)(x)
    x = layers.Dense(256, activation=None)(x)

    return Model(inputs, x)

def build_siamese_model(input_shape):
    input_a = layers.Input(shape=input_shape)
    input_b = layers.Input(shape=input_shape)

    embedding_network = create_embedding_network(input_shape)

    embedding_a = embedding_network(input_a)
    embedding_b = embedding_network(input_b)

    distance = layers.Lambda(
        lambda embeddings: tf.abs(embeddings[0] - embeddings[1])
    )([embedding_a, embedding_b])

    output = layers.Dense(1, activation='sigmoid')(distance)

    return Model(inputs=[input_a, input_b], outputs=output)

input_shape = IMG_SIZE + (1,)
siamese_model = build_siamese_model(input_shape)
siamese_model.summary()

# Compile the model
siamese_model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

# Training
early_stopping = EarlyStopping(monitor='val_loss', patience=10,␣
 ↪restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5,␣
 ↪min_lr=1e-6)
checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss',␣
 ↪save_best_only=True)
```

```
I0000 00:00:1746485560.720979   37181 gpu_device.cc:2022] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 15513 MB memory:  -> device:
0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute capability: 6.0
```

Model: "functional_1"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 128, 128, 1)	0	-
input_layer_1 (InputLayer)	(None, 128, 128, 1)	0	-
functional (Functional)	(None, 256)	1,769,600	input_layer[0][0], input_layer_1[0][0]
lambda (Lambda)	(None, 256)	0	functional[0][0], functional[1][0]
dense_2 (Dense)	(None, 1)	257	lambda[0][0]

Total params: 1,769,857 (6.75 MB)

Trainable params: 1,767,873 (6.74 MB)

Non-trainable params: 1,984 (7.75 KB)

[3]:
```
history = siamese_model.fit(
    [X_train[:, 0], X_train[:, 1]],
    y_train,
    validation_data=([X_val[:, 0], X_val[:, 1]], y_val),
    batch_size=BATCH_SIZE,
    epochs=EPOCHS,
```

```
    callbacks=[early_stopping, reduce_lr, checkpoint]
)

K.clear_session()
tf.compat.v1.reset_default_graph()

# Training Curves
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.savefig('/kaggle/working/training_curves.png')
plt.show()

# Model Evaluation
def evaluate_model(model, X, y, set_name=''):
    batch_size = 8
    y_pred = []
    for i in range(0, len(X), batch_size):
        batch_X = X[i:i + batch_size]
        batch_pred = model.predict([batch_X[:, 0], batch_X[:, 1]],␣
 ↪batch_size=batch_size, verbose=0)
        y_pred.extend(batch_pred)
    y_pred = np.array(y_pred)
    fpr, tpr, thresholds = roc_curve(y, y_pred)
    roc_auc = auc(fpr, tpr)

    plt.figure()
    plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area =␣
 ↪{roc_auc:.2f})')
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlabel('False Positive Rate')
```

```
    plt.ylabel('True Positive Rate')
    plt.title(f'Receiver Operating Characteristic ({set_name})')
    plt.legend(loc="lower right")
    plt.savefig(f'/kaggle/working/roc_curve_{set_name.lower()}.png')
    plt.show()

    return roc_auc

print("Training Evaluation:")
train_auc = evaluate_model(siamese_model, X_train, y_train, 'Training')

print("\nValidation Evaluation:")
val_auc = evaluate_model(siamese_model, X_val, y_val, 'Validation')

print("\nTest Evaluation:")
test_auc = evaluate_model(siamese_model, X_test, y_test, 'Test')

train_acc = history.history['accuracy'][-1]
val_acc = history.history['val_accuracy'][-1]
test_acc = siamese_model.evaluate([X_test[:, 0], X_test[:, 1]], y_test)[1]

print(f"\nFinal Metrics:")
print(f"Training Accuracy: {train_acc:.4f} | AUC: {train_auc:.4f}")
print(f"Validation Accuracy: {val_acc:.4f} | AUC: {val_auc:.4f}")
print(f"Test Accuracy: {test_acc:.4f} | AUC: {test_auc:.4f}")

# Real Photo Evaluation
def visualize_predictions(model, X, y, num_samples=5):
    indices = np.random.choice(len(X), num_samples)
    sample_pairs = X[indices]
    sample_labels = y[indices]

    predictions = model.predict([sample_pairs[:, 0], sample_pairs[:, 1]])

    plt.figure(figsize=(15, 5))
    for i in range(num_samples):
        plt.subplot(2, num_samples, i+1)
        plt.imshow(sample_pairs[i][0].squeeze(), cmap='gray')
        plt.title(f"Label: {sample_labels[i]}\nPred: {predictions[i][0]:.2f}")
        plt.axis('off')

        plt.subplot(2, num_samples, i+1+num_samples)
        plt.imshow(sample_pairs[i][1].squeeze(), cmap='gray')
        plt.axis('off')

    plt.tight_layout()
    plt.show()
```

```
print("Sample Test Predictions:")
visualize_predictions(siamese_model, X_test, y_test)
```

```
Epoch 1/150

WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1746485575.712868   37213 service.cc:148] XLA service 0x7c48e4018ed0
initialized for platform CUDA (this does not guarantee that XLA will be used).
Devices:
I0000 00:00:1746485575.712917   37213 service.cc:156]   StreamExecutor device
(0): Tesla P100-PCIE-16GB, Compute Capability 6.0
I0000 00:00:1746485576.453216   37213 cuda_dnn.cc:529] Loaded cuDNN version
90300

  7/625             13s 22ms/step - accuracy:
0.5285 - loss: 10.5268

I0000 00:00:1746485582.031089   37213 device_compiler.h:188] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.

625/625             37s 37ms/step -
accuracy: 0.8261 - loss: 4.5896 - val_accuracy: 0.7828 - val_loss: 0.7189 -
learning_rate: 0.0010
Epoch 2/150
625/625             14s 22ms/step -
accuracy: 0.9089 - loss: 0.5712 - val_accuracy: 0.9631 - val_loss: 0.6888 -
learning_rate: 0.0010
Epoch 3/150
625/625             14s 22ms/step -
accuracy: 0.9221 - loss: 0.5526 - val_accuracy: 0.8589 - val_loss: 0.7721 -
learning_rate: 0.0010
Epoch 4/150
625/625             14s 22ms/step -
accuracy: 0.9179 - loss: 0.7068 - val_accuracy: 0.6906 - val_loss: 0.9034 -
learning_rate: 0.0010
Epoch 5/150
625/625             14s 22ms/step -
accuracy: 0.9327 - loss: 0.5008 - val_accuracy: 0.9666 - val_loss: 0.5699 -
learning_rate: 0.0010
Epoch 6/150
625/625             14s 22ms/step -
accuracy: 0.9350 - loss: 0.5091 - val_accuracy: 0.9206 - val_loss: 0.6476 -
learning_rate: 0.0010
Epoch 7/150
625/625             14s 22ms/step -
accuracy: 0.9323 - loss: 0.5248 - val_accuracy: 0.9026 - val_loss: 0.7147 -
learning_rate: 0.0010
Epoch 8/150
```

```
625/625            14s 22ms/step -
accuracy: 0.9356 - loss: 0.5196 - val_accuracy: 0.9482 - val_loss: 0.5141 -
learning_rate: 0.0010
Epoch 9/150
625/625            14s 22ms/step -
accuracy: 0.9314 - loss: 0.5314 - val_accuracy: 0.9914 - val_loss: 0.3974 -
learning_rate: 0.0010
Epoch 10/150
625/625            14s 22ms/step -
accuracy: 0.9383 - loss: 0.4247 - val_accuracy: 0.9944 - val_loss: 0.4065 -
learning_rate: 0.0010
Epoch 11/150
625/625            14s 22ms/step -
accuracy: 0.9402 - loss: 0.4597 - val_accuracy: 0.9839 - val_loss: 0.4190 -
learning_rate: 0.0010
Epoch 12/150
625/625            14s 22ms/step -
accuracy: 0.9430 - loss: 0.3902 - val_accuracy: 0.9755 - val_loss: 0.4441 -
learning_rate: 0.0010
Epoch 13/150
625/625            14s 22ms/step -
accuracy: 0.9396 - loss: 0.4103 - val_accuracy: 0.9888 - val_loss: 0.3563 -
learning_rate: 0.0010
Epoch 14/150
625/625            14s 22ms/step -
accuracy: 0.9426 - loss: 0.3571 - val_accuracy: 0.9816 - val_loss: 0.3298 -
learning_rate: 0.0010
Epoch 15/150
625/625            14s 22ms/step -
accuracy: 0.9481 - loss: 0.3355 - val_accuracy: 0.9318 - val_loss: 0.5374 -
learning_rate: 0.0010
Epoch 16/150
625/625            14s 22ms/step -
accuracy: 0.9495 - loss: 0.3512 - val_accuracy: 0.9276 - val_loss: 0.3737 -
learning_rate: 0.0010
Epoch 17/150
625/625            14s 22ms/step -
accuracy: 0.9518 - loss: 0.3182 - val_accuracy: 0.9750 - val_loss: 0.3214 -
learning_rate: 0.0010
Epoch 18/150
625/625            14s 22ms/step -
accuracy: 0.9503 - loss: 0.3084 - val_accuracy: 0.9872 - val_loss: 0.2578 -
learning_rate: 0.0010
Epoch 19/150
625/625            14s 22ms/step -
accuracy: 0.9473 - loss: 0.3168 - val_accuracy: 0.9521 - val_loss: 0.3772 -
learning_rate: 0.0010
Epoch 20/150
```

```
625/625            14s 22ms/step -
accuracy: 0.9535 - loss: 0.2798 - val_accuracy: 0.9512 - val_loss: 0.3468 -
learning_rate: 0.0010
Epoch 21/150
625/625            14s 22ms/step -
accuracy: 0.9496 - loss: 0.2853 - val_accuracy: 0.9755 - val_loss: 0.2846 -
learning_rate: 0.0010
Epoch 22/150
625/625            14s 22ms/step -
accuracy: 0.9517 - loss: 0.2774 - val_accuracy: 0.9780 - val_loss: 0.2664 -
learning_rate: 0.0010
Epoch 23/150
625/625            14s 22ms/step -
accuracy: 0.9604 - loss: 0.2520 - val_accuracy: 0.9841 - val_loss: 0.2181 -
learning_rate: 0.0010
Epoch 24/150
625/625            14s 22ms/step -
accuracy: 0.9635 - loss: 0.2289 - val_accuracy: 0.9890 - val_loss: 0.2252 -
learning_rate: 0.0010
Epoch 25/150
625/625            14s 22ms/step -
accuracy: 0.9626 - loss: 0.2312 - val_accuracy: 0.9575 - val_loss: 0.2847 -
learning_rate: 0.0010
Epoch 26/150
625/625            14s 22ms/step -
accuracy: 0.9546 - loss: 0.2747 - val_accuracy: 0.9736 - val_loss: 0.2640 -
learning_rate: 0.0010
Epoch 27/150
625/625            14s 22ms/step -
accuracy: 0.9647 - loss: 0.2311 - val_accuracy: 0.6929 - val_loss: 0.7050 -
learning_rate: 0.0010
Epoch 28/150
625/625            14s 22ms/step -
accuracy: 0.9529 - loss: 0.2813 - val_accuracy: 0.9902 - val_loss: 0.1842 -
learning_rate: 0.0010
Epoch 29/150
625/625            14s 22ms/step -
accuracy: 0.9639 - loss: 0.2092 - val_accuracy: 0.9907 - val_loss: 0.1913 -
learning_rate: 0.0010
Epoch 30/150
625/625            14s 22ms/step -
accuracy: 0.9667 - loss: 0.2049 - val_accuracy: 0.9916 - val_loss: 0.2100 -
learning_rate: 0.0010
Epoch 31/150
625/625            14s 22ms/step -
accuracy: 0.9687 - loss: 0.2069 - val_accuracy: 0.8786 - val_loss: 0.4246 -
learning_rate: 0.0010
Epoch 32/150
```

```
625/625            14s 22ms/step -
accuracy: 0.9697 - loss: 0.1768 - val_accuracy: 0.9344 - val_loss: 0.3006 -
learning_rate: 0.0010
Epoch 33/150
625/625            14s 22ms/step -
accuracy: 0.9697 - loss: 0.1807 - val_accuracy: 0.9935 - val_loss: 0.1419 -
learning_rate: 0.0010
Epoch 34/150
625/625            14s 22ms/step -
accuracy: 0.9656 - loss: 0.1869 - val_accuracy: 0.9706 - val_loss: 0.2025 -
learning_rate: 0.0010
Epoch 35/150
625/625            14s 22ms/step -
accuracy: 0.9654 - loss: 0.1943 - val_accuracy: 0.9841 - val_loss: 0.1855 -
learning_rate: 0.0010
Epoch 36/150
625/625            14s 22ms/step -
accuracy: 0.9633 - loss: 0.2008 - val_accuracy: 0.9900 - val_loss: 0.1511 -
learning_rate: 0.0010
Epoch 37/150
625/625            14s 22ms/step -
accuracy: 0.9689 - loss: 0.1801 - val_accuracy: 0.9949 - val_loss: 0.1416 -
learning_rate: 0.0010
Epoch 38/150
625/625            14s 22ms/step -
accuracy: 0.9713 - loss: 0.1714 - val_accuracy: 0.9900 - val_loss: 0.1352 -
learning_rate: 0.0010
Epoch 39/150
625/625            14s 22ms/step -
accuracy: 0.9687 - loss: 0.1730 - val_accuracy: 0.9907 - val_loss: 0.1270 -
learning_rate: 0.0010
Epoch 40/150
625/625            14s 22ms/step -
accuracy: 0.9680 - loss: 0.1709 - val_accuracy: 0.9816 - val_loss: 0.1431 -
learning_rate: 0.0010
Epoch 41/150
625/625            14s 22ms/step -
accuracy: 0.9734 - loss: 0.1405 - val_accuracy: 0.9701 - val_loss: 0.1603 -
learning_rate: 0.0010
Epoch 42/150
625/625            14s 22ms/step -
accuracy: 0.9720 - loss: 0.1523 - val_accuracy: 0.9897 - val_loss: 0.1223 -
learning_rate: 0.0010
Epoch 43/150
625/625            14s 22ms/step -
accuracy: 0.9677 - loss: 0.1621 - val_accuracy: 0.9956 - val_loss: 0.1064 -
learning_rate: 0.0010
Epoch 44/150
```

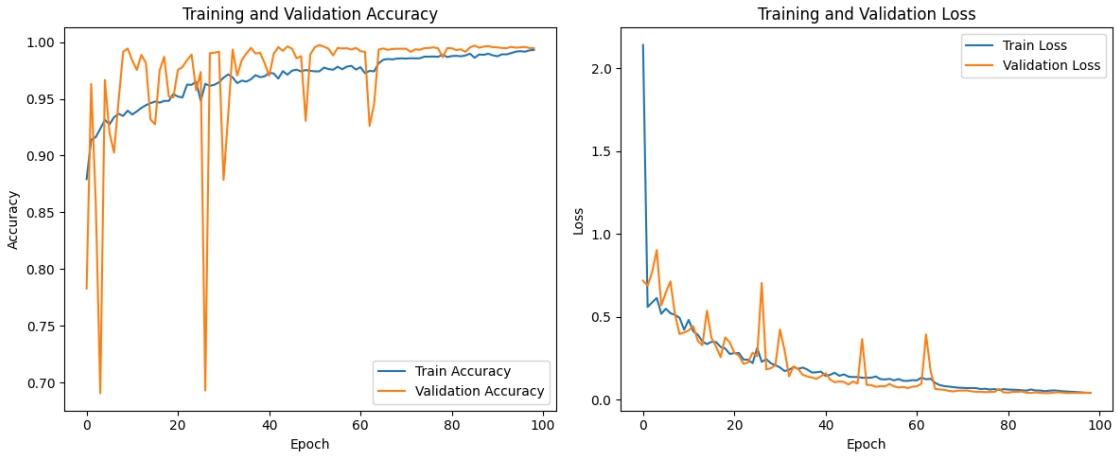
```
625/625            14s 22ms/step -
accuracy: 0.9751 - loss: 0.1458 - val_accuracy: 0.9923 - val_loss: 0.1118 -
learning_rate: 0.0010
Epoch 45/150
625/625            14s 22ms/step -
accuracy: 0.9710 - loss: 0.1498 - val_accuracy: 0.9963 - val_loss: 0.1087 -
learning_rate: 0.0010
Epoch 46/150
625/625            14s 22ms/step -
accuracy: 0.9763 - loss: 0.1399 - val_accuracy: 0.9942 - val_loss: 0.0931 -
learning_rate: 0.0010
Epoch 47/150
625/625            14s 22ms/step -
accuracy: 0.9761 - loss: 0.1374 - val_accuracy: 0.9858 - val_loss: 0.1110 -
learning_rate: 0.0010
Epoch 48/150
625/625            14s 22ms/step -
accuracy: 0.9760 - loss: 0.1339 - val_accuracy: 0.9876 - val_loss: 0.0989 -
learning_rate: 0.0010
Epoch 49/150
625/625            14s 22ms/step -
accuracy: 0.9762 - loss: 0.1285 - val_accuracy: 0.9306 - val_loss: 0.3664 -
learning_rate: 0.0010
Epoch 50/150
625/625            14s 22ms/step -
accuracy: 0.9733 - loss: 0.1372 - val_accuracy: 0.9890 - val_loss: 0.0902 -
learning_rate: 0.0010
Epoch 51/150
625/625            14s 22ms/step -
accuracy: 0.9762 - loss: 0.1240 - val_accuracy: 0.9956 - val_loss: 0.0895 -
learning_rate: 0.0010
Epoch 52/150
625/625            14s 22ms/step -
accuracy: 0.9725 - loss: 0.1475 - val_accuracy: 0.9974 - val_loss: 0.0789 -
learning_rate: 0.0010
Epoch 53/150
625/625            14s 22ms/step -
accuracy: 0.9779 - loss: 0.1244 - val_accuracy: 0.9960 - val_loss: 0.0831 -
learning_rate: 0.0010
Epoch 54/150
625/625            14s 22ms/step -
accuracy: 0.9762 - loss: 0.1251 - val_accuracy: 0.9942 - val_loss: 0.0813 -
learning_rate: 0.0010
Epoch 55/150
625/625            14s 22ms/step -
accuracy: 0.9764 - loss: 0.1268 - val_accuracy: 0.9883 - val_loss: 0.0968 -
learning_rate: 0.0010
Epoch 56/150
```

```
625/625            14s 22ms/step -
accuracy: 0.9800 - loss: 0.1161 - val_accuracy: 0.9949 - val_loss: 0.0817 -
learning_rate: 0.0010
Epoch 57/150
625/625            14s 22ms/step -
accuracy: 0.9752 - loss: 0.1256 - val_accuracy: 0.9944 - val_loss: 0.0752 -
learning_rate: 0.0010
Epoch 58/150
625/625            14s 22ms/step -
accuracy: 0.9795 - loss: 0.1162 - val_accuracy: 0.9946 - val_loss: 0.0779 -
learning_rate: 0.0010
Epoch 59/150
625/625            14s 22ms/step -
accuracy: 0.9802 - loss: 0.1158 - val_accuracy: 0.9935 - val_loss: 0.0712 -
learning_rate: 0.0010
Epoch 60/150
625/625            14s 22ms/step -
accuracy: 0.9758 - loss: 0.1177 - val_accuracy: 0.9949 - val_loss: 0.0797 -
learning_rate: 0.0010
Epoch 61/150
625/625            14s 22ms/step -
accuracy: 0.9793 - loss: 0.1161 - val_accuracy: 0.9921 - val_loss: 0.0819 -
learning_rate: 0.0010
Epoch 62/150
625/625            14s 22ms/step -
accuracy: 0.9738 - loss: 0.1346 - val_accuracy: 0.9914 - val_loss: 0.0982 -
learning_rate: 0.0010
Epoch 63/150
625/625            14s 22ms/step -
accuracy: 0.9740 - loss: 0.1259 - val_accuracy: 0.9262 - val_loss: 0.3948 -
learning_rate: 0.0010
Epoch 64/150
625/625            14s 22ms/step -
accuracy: 0.9728 - loss: 0.1336 - val_accuracy: 0.9456 - val_loss: 0.1749 -
learning_rate: 0.0010
Epoch 65/150
625/625            14s 22ms/step -
accuracy: 0.9802 - loss: 0.1114 - val_accuracy: 0.9935 - val_loss: 0.0664 -
learning_rate: 2.0000e-04
Epoch 66/150
625/625            14s 22ms/step -
accuracy: 0.9854 - loss: 0.0894 - val_accuracy: 0.9944 - val_loss: 0.0625 -
learning_rate: 2.0000e-04
Epoch 67/150
625/625            14s 22ms/step -
accuracy: 0.9847 - loss: 0.0845 - val_accuracy: 0.9932 - val_loss: 0.0614 -
learning_rate: 2.0000e-04
Epoch 68/150
```

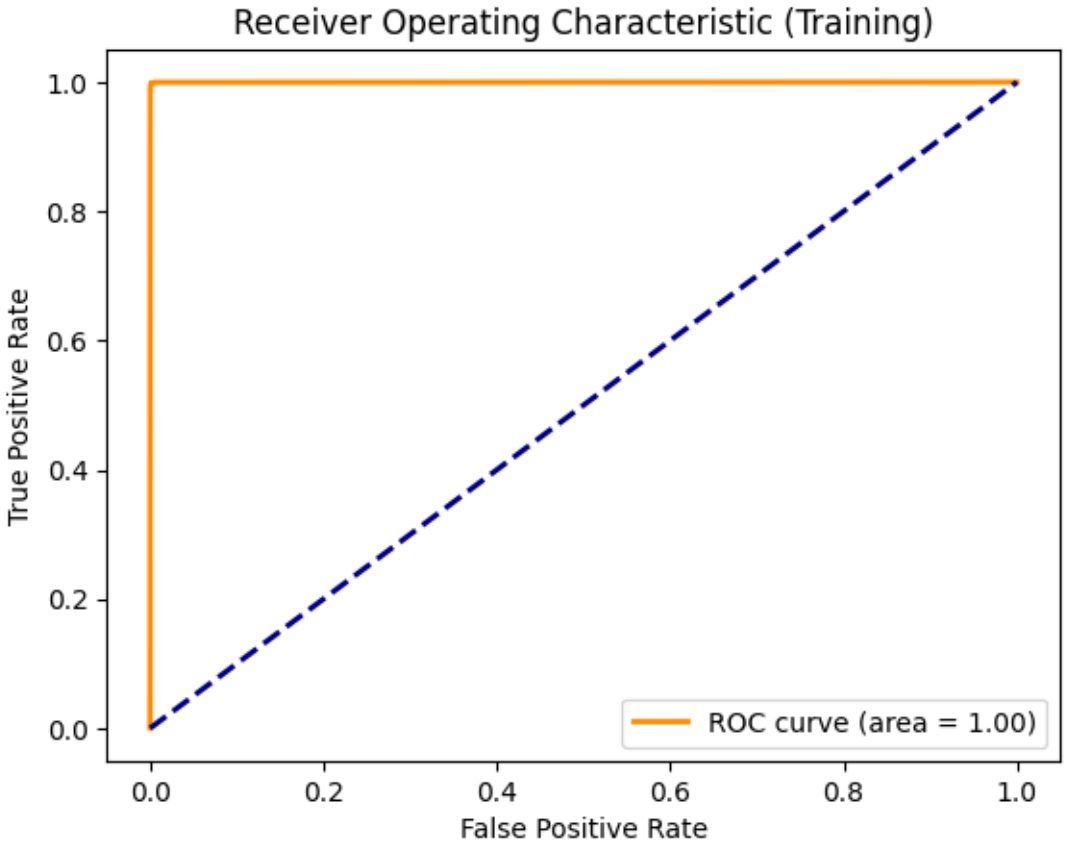
```
625/625             14s 22ms/step -
accuracy: 0.9840 - loss: 0.0823 - val_accuracy: 0.9939 - val_loss: 0.0544 -
learning_rate: 2.0000e-04
Epoch 69/150
625/625             14s 22ms/step -
accuracy: 0.9857 - loss: 0.0768 - val_accuracy: 0.9942 - val_loss: 0.0512 -
learning_rate: 2.0000e-04
Epoch 70/150
625/625             14s 22ms/step -
accuracy: 0.9857 - loss: 0.0741 - val_accuracy: 0.9942 - val_loss: 0.0557 -
learning_rate: 2.0000e-04
Epoch 71/150
625/625             14s 22ms/step -
accuracy: 0.9846 - loss: 0.0744 - val_accuracy: 0.9942 - val_loss: 0.0559 -
learning_rate: 2.0000e-04
Epoch 72/150
625/625             14s 23ms/step -
accuracy: 0.9862 - loss: 0.0722 - val_accuracy: 0.9914 - val_loss: 0.0566 -
learning_rate: 2.0000e-04
Epoch 73/150
625/625             14s 22ms/step -
accuracy: 0.9853 - loss: 0.0737 - val_accuracy: 0.9937 - val_loss: 0.0516 -
learning_rate: 2.0000e-04
Epoch 74/150
625/625             14s 22ms/step -
accuracy: 0.9856 - loss: 0.0733 - val_accuracy: 0.9932 - val_loss: 0.0490 -
learning_rate: 2.0000e-04
Epoch 75/150
625/625             14s 22ms/step -
accuracy: 0.9865 - loss: 0.0662 - val_accuracy: 0.9946 - val_loss: 0.0491 -
learning_rate: 2.0000e-04
Epoch 76/150
625/625             14s 22ms/step -
accuracy: 0.9865 - loss: 0.0685 - val_accuracy: 0.9949 - val_loss: 0.0470 -
learning_rate: 2.0000e-04
Epoch 77/150
625/625             14s 22ms/step -
accuracy: 0.9873 - loss: 0.0630 - val_accuracy: 0.9953 - val_loss: 0.0482 -
learning_rate: 2.0000e-04
Epoch 78/150
625/625             14s 22ms/step -
accuracy: 0.9874 - loss: 0.0648 - val_accuracy: 0.9946 - val_loss: 0.0487 -
learning_rate: 2.0000e-04
Epoch 79/150
625/625             14s 22ms/step -
accuracy: 0.9891 - loss: 0.0583 - val_accuracy: 0.9867 - val_loss: 0.0653 -
learning_rate: 2.0000e-04
Epoch 80/150
```

```
625/625            14s 22ms/step -
accuracy: 0.9853 - loss: 0.0685 - val_accuracy: 0.9946 - val_loss: 0.0452 -
learning_rate: 2.0000e-04
Epoch 81/150
625/625            14s 22ms/step -
accuracy: 0.9890 - loss: 0.0591 - val_accuracy: 0.9946 - val_loss: 0.0440 -
learning_rate: 2.0000e-04
Epoch 82/150
625/625            14s 22ms/step -
accuracy: 0.9878 - loss: 0.0623 - val_accuracy: 0.9930 - val_loss: 0.0482 -
learning_rate: 2.0000e-04
Epoch 83/150
625/625            14s 22ms/step -
accuracy: 0.9875 - loss: 0.0616 - val_accuracy: 0.9937 - val_loss: 0.0485 -
learning_rate: 2.0000e-04
Epoch 84/150
625/625            14s 22ms/step -
accuracy: 0.9888 - loss: 0.0574 - val_accuracy: 0.9914 - val_loss: 0.0505 -
learning_rate: 2.0000e-04
Epoch 85/150
625/625            14s 22ms/step -
accuracy: 0.9902 - loss: 0.0560 - val_accuracy: 0.9953 - val_loss: 0.0433 -
learning_rate: 2.0000e-04
Epoch 86/150
625/625            14s 22ms/step -
accuracy: 0.9850 - loss: 0.0660 - val_accuracy: 0.9970 - val_loss: 0.0421 -
learning_rate: 2.0000e-04
Epoch 87/150
625/625            14s 22ms/step -
accuracy: 0.9886 - loss: 0.0570 - val_accuracy: 0.9951 - val_loss: 0.0458 -
learning_rate: 2.0000e-04
Epoch 88/150
625/625            14s 22ms/step -
accuracy: 0.9897 - loss: 0.0560 - val_accuracy: 0.9960 - val_loss: 0.0424 -
learning_rate: 2.0000e-04
Epoch 89/150
625/625            14s 23ms/step -
accuracy: 0.9895 - loss: 0.0541 - val_accuracy: 0.9965 - val_loss: 0.0408 -
learning_rate: 2.0000e-04
Epoch 90/150
625/625            14s 22ms/step -
accuracy: 0.9882 - loss: 0.0561 - val_accuracy: 0.9956 - val_loss: 0.0408 -
learning_rate: 2.0000e-04
Epoch 91/150
625/625            14s 22ms/step -
accuracy: 0.9871 - loss: 0.0594 - val_accuracy: 0.9953 - val_loss: 0.0443 -
learning_rate: 2.0000e-04
Epoch 92/150
```

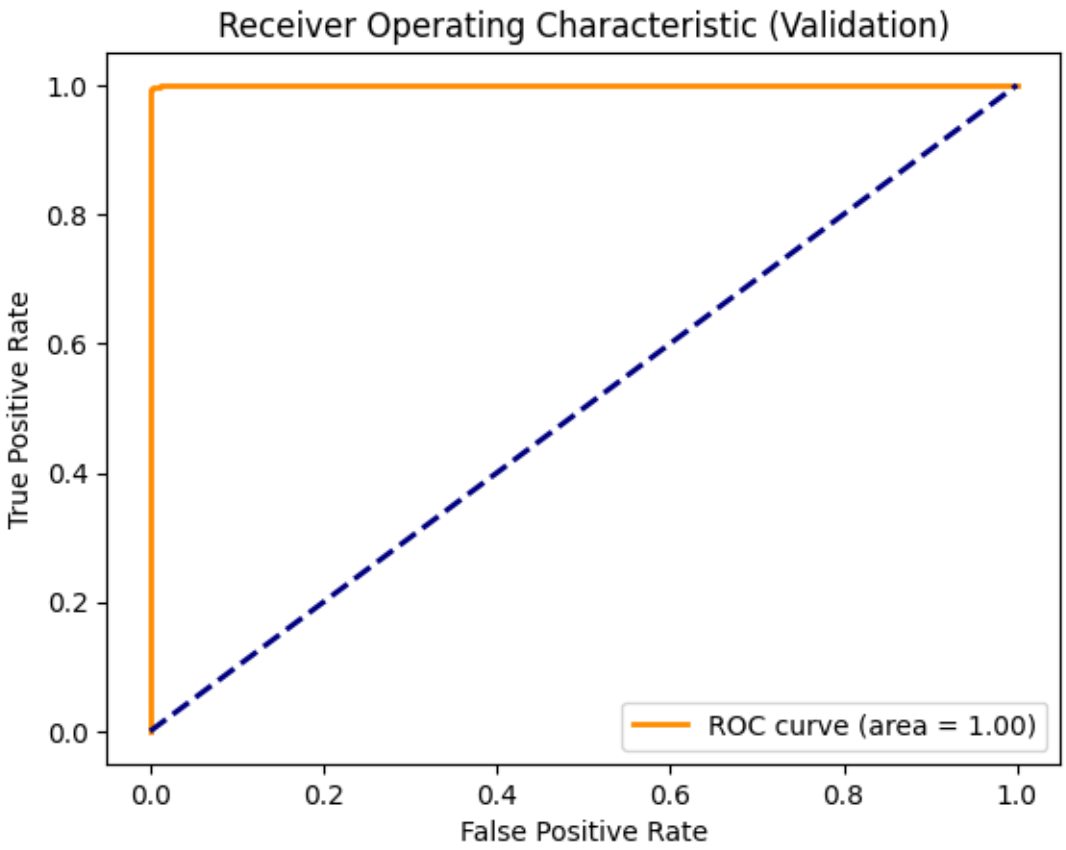
```
625/625             14s 22ms/step -
accuracy: 0.9892 - loss: 0.0553 - val_accuracy: 0.9949 - val_loss: 0.0462 -
learning_rate: 2.0000e-04
Epoch 93/150
625/625             14s 22ms/step -
accuracy: 0.9884 - loss: 0.0561 - val_accuracy: 0.9946 - val_loss: 0.0435 -
learning_rate: 2.0000e-04
Epoch 94/150
625/625             14s 22ms/step -
accuracy: 0.9905 - loss: 0.0483 - val_accuracy: 0.9958 - val_loss: 0.0409 -
learning_rate: 2.0000e-04
Epoch 95/150
625/625             14s 22ms/step -
accuracy: 0.9914 - loss: 0.0509 - val_accuracy: 0.9951 - val_loss: 0.0423 -
learning_rate: 4.0000e-05
Epoch 96/150
625/625             14s 22ms/step -
accuracy: 0.9920 - loss: 0.0477 - val_accuracy: 0.9953 - val_loss: 0.0419 -
learning_rate: 4.0000e-05
Epoch 97/150
625/625             14s 22ms/step -
accuracy: 0.9909 - loss: 0.0470 - val_accuracy: 0.9958 - val_loss: 0.0419 -
learning_rate: 4.0000e-05
Epoch 98/150
625/625             14s 22ms/step -
accuracy: 0.9927 - loss: 0.0448 - val_accuracy: 0.9949 - val_loss: 0.0429 -
learning_rate: 4.0000e-05
Epoch 99/150
625/625             14s 22ms/step -
accuracy: 0.9932 - loss: 0.0417 - val_accuracy: 0.9946 - val_loss: 0.0432 -
learning_rate: 4.0000e-05
```



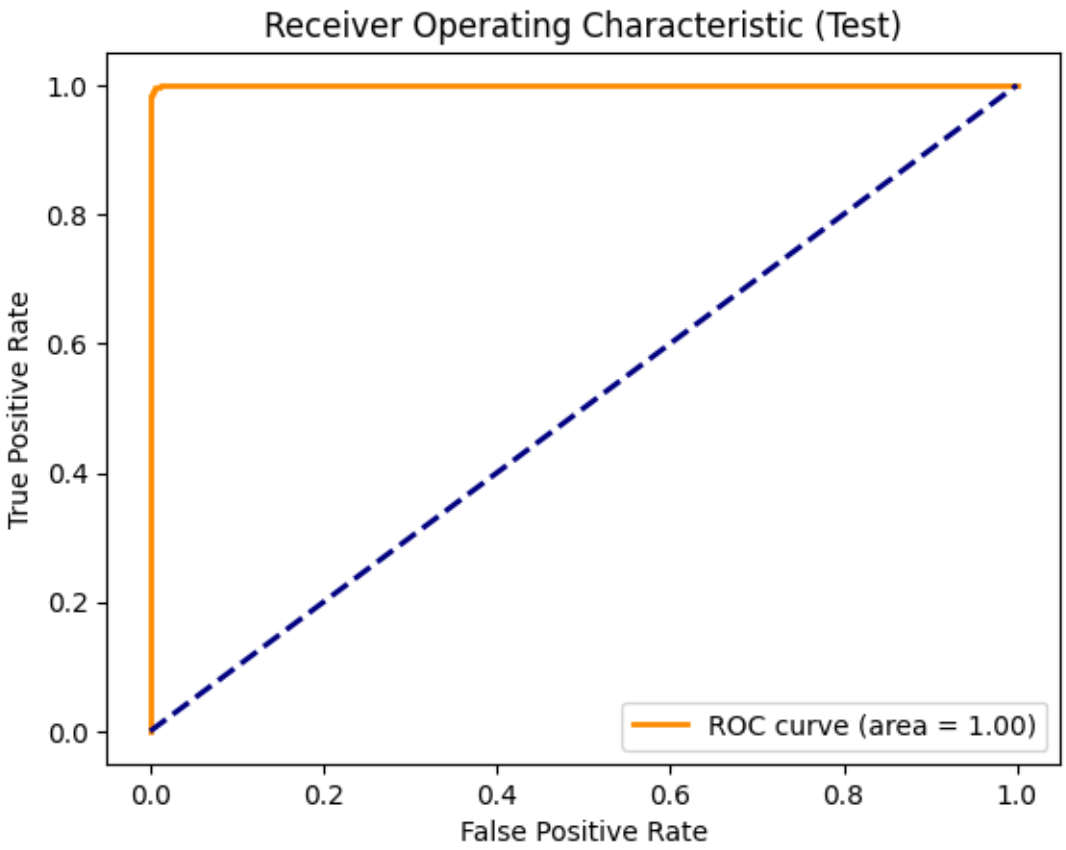
Training Evaluation:



Validation Evaluation:




```
Test Evaluation:
```




```
134/134             1s 6ms/step -
accuracy: 0.9935 - loss: 0.0430

Final Metrics:
Training Accuracy: 0.9931 | AUC: 1.0000
Validation Accuracy: 0.9946 | AUC: 0.9999
Test Accuracy: 0.9953 | AUC: 0.9999
Sample Test Predictions:
1/1             1s 608ms/step
```


Label: 1
Pred: 0.97
Label: 0
Pred: 0.00
Label: 0
Pred: 0.00
Label: 0
Pred: 0.00
Label: 0
Pred: 0.00